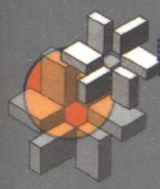
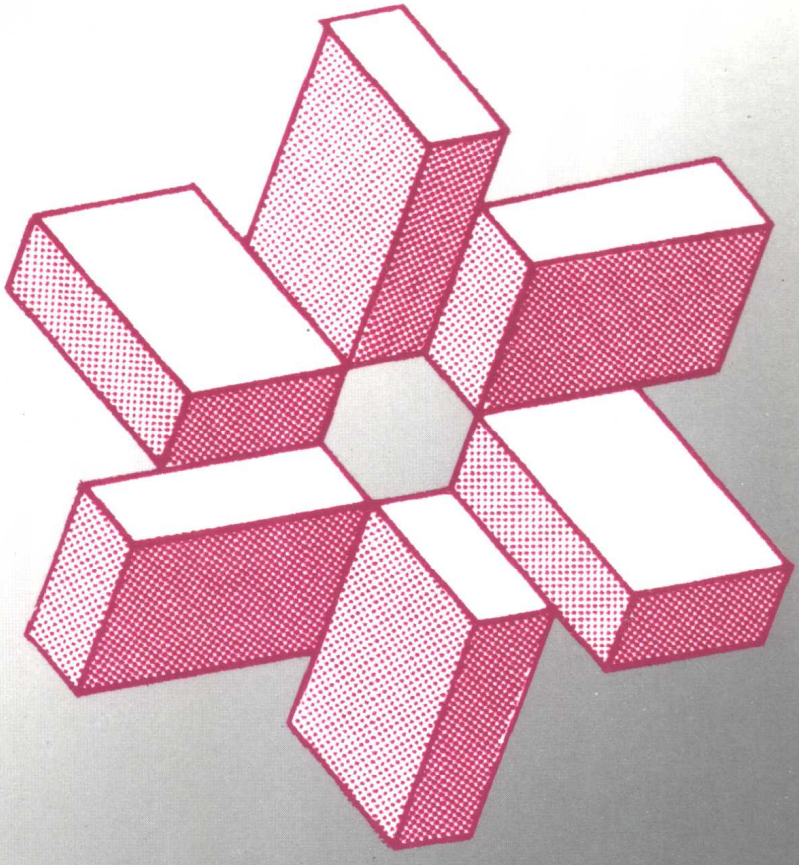


看实例学编程系列丛书

Delphi 5 API

函数开发实例

同志工作室　编著



人民邮电出版社
www.pptph.com.cn

看实例学编程系列丛书

Delphi 5 API 函数开发实例

同志工作室　　编著

人 民 邮 电 出 版 社

内 容 提 要

Delphi 5 是美国 Inprise 公司开发的，运行于 Windows 平台上的交互式可视化集成开发环境。本书从 API 编程基础开始，以示例的形式全面介绍了 Delphi 5 支持的 API 函数以及 API 函数的应用，涵盖了文本、图形、高级绘图、图像处理、窗口、菜单、系统信息控制、消息控制等各个方面，揭去了 API 函数的神秘面纱，带领读者进入 Windows 程序开发的内部。

本书通俗易懂，示例丰富，讲解细致，分析透彻，适合于中级程序开发人员学习使用，对于从事 API 函数开发与应用的广大科研人员、高校相关专业的师生也是一本有价值的自学和教学的参考书。

看实例学编程系列丛书

Delphi 5 API 函数开发实例

◆ 编　　著　同志工作室

责任编辑　姚予疆

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号

邮编　100061　　电子函件　315@ pptph.com.cn

网址　http://www.pptph.com.cn

北京鸿佳印刷厂印刷

新华书店总店北京发行所经销

◆ 开本：787×1092　1/16

印张：19.75

字数：493 千字　　2001 年 1 月第 1 版

印数：4 001－7 000 册　　2001 年 2 月北京第 2 次印刷

ISBN 7-115-06908-5/TP·597

定价：29.00 元

编 者 的 话

面向对象技术近年来发展迅速，它被广泛地应用到计算机研究与应用的各个方面，如文件处理、操作系统设计、多媒体技术、网络与数据库开发等。用面向对象技术进行程序设计、开发软件已经成为一种时尚。这种技术从根本上改变了人们以往设计软件的思维方式，从而使程序设计者可以最大限度地摆脱烦琐的数据格式和冗长的研发过程，将精力集中在对要处理的对象的设计和研究上，大大提高了软件开发的效率。为了满足初中级 Windows 程序开发人员、大专院校相关专业师生及业余爱好者学习和应用各种流行程序设计软件的需求，我们同志工作室经过多方调研，在收集了不同层次读者意见的基础上，经过仔细研讨，于 2000 年 5 月份推出了《看实例学编程》系列丛书的前 5 本。

《看实例学编程》系列丛书介绍的软件都是国内外著名软件公司的知名产品，也是国内应用面最广的软件。本套丛书一改以往计算机编程图书枯燥的风格，将软件开发技术融合到程序示例中，采用了由实际到理论、由具体到抽象的逆向写作思路。丛书一经推出，就立即得到了广大读者的好评，同时，也有不少读者建议，能不能以这种方式更深入地介绍软件开发的各专项领域。为了满足广大读者的需求，我们同志工作室的全体成员经过多方讨论，又精心策划了下面 10 本专题类图书，收入本套丛书内。它们是：《Delphi 5 数据库开发实例》、《Visual Basic 6.0 数据库开发实例》、《Visual C++ 6.0 数据库开发实例》、《C++ Builder 5.0 数据库开发实例》、《Delphi 5 API 函数开发实例》、《Visual Basic 6.0 API 函数开发实例》、《C++ Builder 5.0 API 函数开发实例》、《Delphi 5 多媒体开发实例》、《Visual Basic 6.0 多媒体开发实例》及《C++ Builder 5.0 多媒体开发实例》。

这 10 本书秉承了前 5 本书的特点，但它更侧重于软件开发的具体领域。例如，数据库、多媒体和 API 函数，而不是广泛地学习软件各个方面的知识；不是繁琐冗长的使用手册或枯燥乏味的大本参考书，而是独具实效的实例指南。这 10 本书准确地告诉读者用程序设计软件可以做哪些开发工作以及如何做这些开发工作，内容充实、讲解细致、分析透彻，笔调亲切，绝没有居高临下的架势。而且，我们在编写的过程中尽量省去了枯燥难懂的专业术语，以平和易懂的语言带领大家逐步进入到编程的艺术天堂。这些书以计算机中级程序开发人员为主要的读者对象，为便于读者理解，我们根据自己学习和使用的体会精心挑选了大量的实例，这些实例都是针对程序员在开发过程中最需掌握的技术而特意定制的，能较好地满足读者的需求。

同志工作室

前　言

Delphi 5 是美国 Inprise 公司开发的，运行于 Windows 平台上的交互式可视化集成开发环境。像其他的可视化集成开发环境（如 Visual Basic、Visual C++）一样，Delphi 5 集程序的代码编辑、编译、连接及调试等功能于一体，给编程人员提供了一个完整、方便的开发界面和许多有效的辅助开发工具。Delphi 5 的应用程序向导可以为很大一部分的程序提供框架代码，用户不需书写代码，只要按几个按钮就可以生成一个完整的可以运行的程序。

本书通过丰富的示例程序向读者介绍如何使用 Delphi 5 支持的 API 函数开发 Windows 应用程序。它的特点在于使用了大量的示例程序，这些程序都是笔者根据自己学习和使用中的体会精心编写的，是针对程序员在开发过程中最迫切需要、使用频率最高的内容特意定制的，可以说比较贴切地符合了初级和中级程序员的需求。另外，本书中所有示例程序都经过了严格的调试和测试，读者只要跟着书中给出的步骤往下做，最终一定能够圆满地完成程序。

第 1 章首先介绍了与文本有关的 API 函数。Delphi 5 本身不能实现一行字符的倾斜，在本章的旋转字体程序中，读者可以初步领略到 API 函数的神奇。

在 Delphi 5 中提供了一些绘图的方法，API 函数也有相应的绘图函数，并且可以实现更强大的功能。用 Delphi 5 支持的 API 函数，配合其他的一些函数，完全可以实现像 Windows “附件”——画图那样的程序，甚至可以做得更好。本章通过示例程序向读者说明了利用 API 函数绘制图形的技术。

在接触 API 函数之前，我们不能够定制绘图环境，仅仅能够绘制基本图形。通过第 2 章的学习，读者可以更加灵活地处理图形应用程序，本章综合利用 API 各个绘图函数制作了很多示例程序，希望读者认真体会，从中领略到 API 函数强大的功能。

第 3 章主要介绍了创建位图、装载位图、设置鼠标形状、创建图标、装载图标和图像处理函数。其中，图像的处理比较常用，希望读者仔细阅读本章，掌握用 API 函数处理图像的技术。

第 4 章介绍了文件的创建、打开、修改和关闭，系统目录和驱动器信息的取得、更改和创建以及有关注册表的操作。本章内容较多，比较抽象，可以加深读者对 Windows 操作系统的了解。

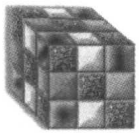
第 5 章介绍了控制窗体和菜单的 API 函数，主要内容有窗体之间的关系、排列窗口、获得及改变窗口的状态、窗体与矩形、获得菜单属性、添加与删除菜单和设置菜单等。窗口和菜单是 Windows 操作系统中重要的部件，在试着使用这些函数时，是不是有一种恍然大悟的感觉呢？

第 6 章介绍了与 Windows 系统有关的函数和消息，包括鼠标信息的读取和设置、剪贴板操作、系统信息的读取和设置以及消息传递，通过这些函数和消息，读者可以对 Windows 进行一些简单地定制。当然，如果能够把功能强大的 API 函数与简单易用的 Delphi 5 结合起来的话，一定能够使编程水平更上一层楼。

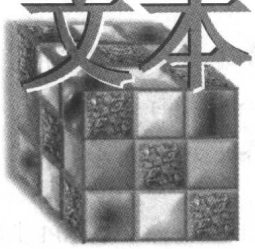
本书由张智慧、郭燕、金荣、尹玉和潘利华等同志编写。由于编写时间紧张，作者水平有限，书中难免存在一些不足之处，恳请读者批评指正。

编著者

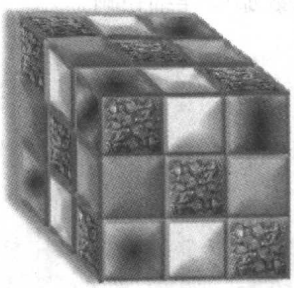
目　　录

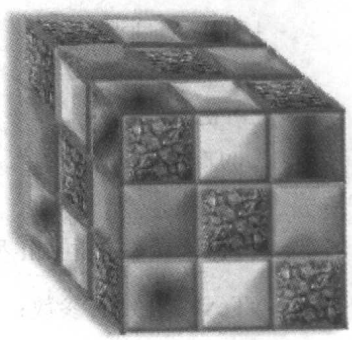


第1章 文本与图形绘制



- 文本处理
- 画线函数
- 绘制多边形
- 绘制矩形
- 绘制椭圆、弧、弦





文本处理

对文本的处理是在程序中最常用到的，API 函数提供了非常强大的文本处理功能，调用 API 函数，可以实现 Delphi 5 自身无法实现的功能。

这一节主要介绍以下函数以及在它们的参数中涉及到的结构定义：

- DrawText(): 将文本描绘到指定的矩形中；
- DrawTextEx(): 与 DrawText()函数相似，只是加入了更多的功能；
- GetTextColor(): 获得当前文本颜色，文本颜色通常也称为“前景色”；
- SetTextColor(): 设置当前文本颜色；
- TextOut(): 输出文本函数；
- ExtTextOut(): 经过扩展的输出文本函数；
- GetTextAlign(): 获得指定设备当前的文本对齐标志；
- SetTextAlign(): 设置指定设备的文本对齐方式。

在本书的函数介绍中，你会经常看到“设备”这个词，“设备”在 API 函数中指窗体、控件等。

下面分别对上述函数加以详细讲解。

DrawText()

DrawText()函数用于将文本描绘到指定的矩形中。

它的声明形式如下所示：

```
int DrawText
(
    HDC hdc,
    LPCTSTR lpString,
    int nCount,
    LPRECT lpRect,
    UINT uFormat
);
```

DrawText()函数返回 int 型值，代表描绘的文字的高度。

DrawText()函数的参数：

- hdc：HDC 型，要显示文字的设备的句柄；
- lpString：LPCTSTR 型，指向存储着要显示的文字的字符串；
- nCount：int 型，要描绘的字符数量，如果想要描绘整个字符串（直到终止符），则可将这个参数设为-1；
- lpRect：LPRECT 型，指向一个定义了矩形的 RECT 结构（采用逻辑坐标）；
- uFormat：UINT 型，一个标志位数组，决定了以何种形式执行绘图。

uFormat 参数值可以是以下常数之一或它们的不矛盾的组合：
DT_BOTTOM：指示文本对齐矩形的底边，必须与 DT_SINGLELINE 联合使用；
DT_TOP：指示文本对齐到矩形的顶部，必须与 DT_SINGLELINE 联合使用；
DT_SINGLELINE：绘制单行文本；
DT_CALCRECT：有多行文字时，矩形的底边根据需要进行延展，以便容下所有文字；单行绘制文本时，延展矩形的右侧；
DT_EXPANDTABS：描绘文字的时候，对制表站进行扩展，默认的制表站间距是 8 个字符；
DT_EXTERNALLEADING：计算文本行高度的时候，使用当前字体的外部间距属性（the external leading attribute）；
DT_LEFT：文本左对齐；
DT_RIGHT：文本右对齐；
DT_CENTER：文本水平居中；
DT_VCENTER：文本垂直居中，必须与 DT_SINGLELINE 联合使用；
DT_NOPREFIX：通常，函数认为&字符表示应为下一个字符加上下划线，该标志禁止这种行为；
DT_WORDBREAK：进行自动换行，如果用 SetTextAlign 函数设置了 TA_UPDATECP 标志，这里的设置则无效；
DT_NOCLIP：不加剪裁地显示；

DrawText()函数使用选定的字体、前景色和背景色来显示文本，除非指定了 DT_NOCLIP 格式，否则，DrawText()会剪裁文本使之不显示在定义的矩形外；除非指定了 DT_SINGLELINE 格式，否则，所有的格式化都是针对多行。

RECT 结构的定义如下所示：

```
typedef struct _RECT
{
    long left;
    long top;
    long right;
    long bottom;
} RECT;
```

RECT 结构定义了一个矩形的左上角和右下角的坐标，以逻辑坐标表示。

从上面这个函数的声明形式和参数介绍中，读者是否注意到，有一些比较陌生的数据类型，例如，HDC、LPCTSTR 及 UINT，这些以大写字母表示的数据类型，是复合型的数据类型：

- HDC：表示设备（或对象）的句柄；

还有一些以 H 开头的类型，例如，HBrush 表示一个画刷的句柄，HWnd 表示窗口的句柄，HMenu 表示菜单的句柄等，它们均为 Long 型。

- LPCTSTR：以 LP(Long Pointer 的缩写) 开头，表示为一个指针类型，STR 表示它是一个指向字符串的指针；

还有一些以 LP 开头的类型，它们都是指针类型，LP 后面的字符表明了它们指向哪一类数据。

- UINT：无符号整数。

DrawTextEx()

与 DrawText()函数相似，只是加入了更多的功能，例如，添加了更多的文本在矩形中显示的格式以及附加的绘图参数。

它的声明形式如下所示：

```
int DrawTextEx
(
    HDC hdc,
    LPCTSTR lpchText,
    int cchText,
    LPRECT lprc,
    UINT dwDTFormat,
    LPDRAWTEXTPARAMS lpDTParams
);
```

DrawTextEx()函数返回 int 型值，代表描绘的文字的高度。

DrawTextEx()函数的参数：

- hdc：HDC 型，要在其中显示文字的设备的句柄；
- lpchText：LPCTSTR 型，指向存储着要显示文字的字符串；
- cchText：int 型，要描绘的字符数量，如果要描绘整个字符串（直到终止符），则可将这个参数设为-1；
- lprc：LPRECT 型，指定一个矩形，文本在矩形范围内；
- dwDTFormat：UINT 型，一个标志位，决定了以何种形式执行文本的显示，参考 DrawText()的 uFormat 参数和下面的常数；

下面列出的是 dwDTFormat 参数值可选择的新增的常数：

DT_EDITCONTROL：模拟一个多行编辑控件，不显示部分可见的行；
DT_END_ELLIPSES：倘若字符串不能在矩形里全部容下，就在末尾显示省略号；
DT_PATH_ELLIPSES：倘若字符串不能在矩形里全部容下，可以在字符串中包含 \ 字符，就会用省略号替换这个字符串内容，例如，一个很长的路径名可能显示成 c:\windows\...\doc\readme.txt；
DT_MODIFYSTRING：如果指定了 DT_END_ELLIPSES 或 DT_PATH_ELLIPSES，就会对字符串进行修改，使其与实际显示的字符串相符；
DT_RTLREADING：如果选入设备场景的字体属于希伯来或阿拉伯语系，就从右到左描绘文字。

- lpDTParams：LPDRAWTEXTPARAMS 结构，这个结构包含了附加的绘图参数，lpDTParams 参数可以为空。

下面来分析一下 DRAWTEXTPARAMS 结构，它的定义如下所示：

```
typedef struct {
    UINT cbSize;
    int   iTabLength;
    int   iLeftMargin;
    int   iRightMargin;
    UINT uiLengthDrawn;
} DRAWTEXTPARAMS, FAR *LPDRAWTEXTPARAMS;
```

其中，各成员的说明如下：

- cbSize：指定结构的大小；
- iTabLength：设置 Tab 键所占的大小，以字符的平均宽度为单位；
- iLeftMargin：设置左边距，以字符的平均宽度为单位；
- iRightMargin：设置右边距，以字符的平均宽度为单位；
- uiLengthDrawn：接收 DrawTextEx()函数描绘的字符个数，包括空格。

GetTextColor()

GetTextColor()函数用于返回指定的设备中当前文本的颜色，也就是前景色。

它的声明形式如下所示：

```
COLORREF GetTextColor
(
    HDC hdc
);
```

GetTextColor()函数返回 COLORREF 型值，代表文本的当前颜色设置；如果出错，返回 CLR_INVALID。

GetTextColor()函数的参数只有一个——hdc，HDC 型，代表设备的句柄。

SetTextColor()

SetTextColor()函数用于设置当前文本颜色。

它的声明形式如下所示：

```
COLORREF SetTextColor
(
    HDC hdc,
    COLORREF crColor
);
```

SetTextColor()函数返回 COLORREF 型值，代表设置前的颜色设定；如果返回值为CLR_INVALID，表示调用失败。

SetTextColor()函数的参数：

- hdc：HDC 型，要设置文本颜色的设备的句柄；
- crColor：COLORREF 型，新的文本颜色。

COLORREF 型数值为 32 位整数，用来表示颜色的 R（红）、G（绿）、B（蓝）颜色分量值。

示例程序

示例程序代码如下所示：

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type
  TForm1 = class(TForm)
    procedure FormMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;
  TheRect:TRect;
  hdc1:HDC;
  str1:PChar;

implementation
{$R *.DFM}

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    TheRect.Left:=X;
    TheRect.Top:=Y;
    TheRect.Right:=X+100;
    TheRect.Bottom:=Y+100;
    hdc1:=GetDC(self.Handle);
    SetTextColor(hdc1,RGB(255,0,0));
    SetBkMode(hdc1,TRANSPARENT);
    str1:='同志工作室';
    DrawText(hdc1,str1,-1,TheRect,DT_LEFT);
end;
end.
```

程序运行结果如图 1-1 所示。

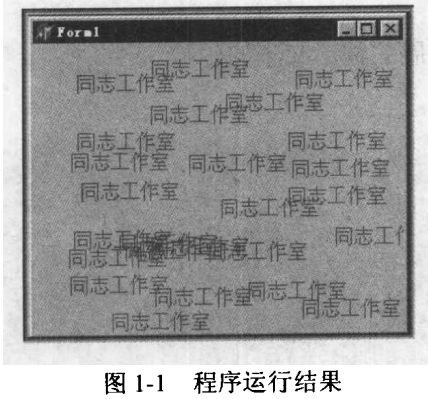


图 1-1　程序运行结果

TextOut()

TextOut()函数用于在指定设备(窗体或控件等)上输出指定的文本。

它的声明形式如下所示：

```
BOOL TextOut
(
    HDC hdc,
    int nXStart,
    int nYStart,
    LPCTSTR lpString,
    int cbString
);
```

TextOut()函数返回 BOOL 型值，如果返回值为 False，表示调用失败；返回值为 True，表示调用成功。

TextOut()函数的参数：

- hdc：HDC 型，表示设备的句柄；
- nXStart 和 nYStart：int 型，要描绘的文本的起始点坐标；
- lpString：LPCTSTR 型，指向存储要描绘的文本的字符串；
- cbString：int 型，字符串中要描绘的字符数量。

如果绘图背景模式是“不透明”（opaque）的，那么创建的轮廓将由字符单元格减去字符构成；如背景模式为“透明”，轮廓就由字符的字样本身构成。

ExtTextOut()

ExtTextOut()函数是经过扩展的输出文本的函数。

它的声明形式如下所示：

```
BOOL ExtTextOut
(
    HDC hdc,
    int X,
    int Y,
    UINT fuOptions,
    CONST RECT *lprc,
```